

Contents

| | |
|--|----------|
| 1 Routine/Function Prologues | 2 |
| 1.1 Fortran: Module Interface FVadvcore_GridCompMod (Source File: FVadvcore_GridCompMod.F90) | 2 |
| 1.1.1 SetServices | 3 |
| 1.1.2 Initialize | 5 |
| 1.1.3 Run | 5 |
| 1.1.4 Finalize | 6 |
| 1.1.5 TestCase | 7 |

1 Routine/Function Prologues

1.1 Fortran: Module Interface FVadvcore_GridCompMod (Source File: FVadvcore_GridCompMod.F90)

This a MAPL component that can be used in either with offline or online applications to advect an arbitrary set of constituents.

Scientific Description: The advection scheme used is that from the FVdycore grid-point dynamical core. It runs on a sphere and uses finite-volume discretization techniques. The advection is time split into a horizontal phase that is assumed to be vertically Lagrangian and a vertical remap phase. A complete description of the core from which this component is taken may be found in:

Lin, S.-J. 2004, A vertically Lagrangian Finite-Volume Dynamical Core for Global Models. *Mon. Wea. Rev.*, **132**, 2293-2307.

Code Implementation: The code uses the MAPL (<http://MAPLCode.org/maplwiki/>) to encapsulate the FV advection scheme as an ESMF gridded component using the ESMF paradigm of initialize, run and finalize methods, and their SetServices routine. As in all ESMF codes, only SetServices is public and the interface consists of a Clock and Import and Export states. The import state includes a specialized description of the motion field in terms of C-grid winds and mass fluxes. These are assumed to have been accumulated over the time interval specified in the resource file. The default of this interval is 1800 seconds. The layer pressure thicknesses in the import state are assumed to be the instantaneous values valid at the beginning of this interval. If these thicknesses are friendly they will be updated to values valid at the end of the interval, consistent with the given motion field. Mixing ratios of the constituents to be advected are placed ESMF Fields within an ESMF Bundle in the Import state. Each Field in the Bundle is tested for “Friendliness” to advection; if friendly it is advected and its values updated.

Currently no Export capability is implemented.

INTERFACE:

```
module FVadvcore_GridCompMod
```

USES:

```
use ESMF_Mod
use MAPL_Mod

use TracerMod,      only : T_TRACERS, Tracer_Get, Tracer_Set
use FVAdvMod,       only : T_FVAdv_STATE, FVAdvReal, &
                           FVAdv_Init,   FVAdv_Run, FVAdv_Final, &
                           FVAdv_Remap, FVAdv_PrepVars
implicit none
private
```

PUBLIC MEMBER FUNCTIONS:

```
public SetServices
```

1.1.1 SetServices - Externally visible registration routine

INTERFACE:

```
subroutine SetServices(GC, rc)
```

ARGUMENTS:

```
type(ESMF_GridComp), intent(inout) :: GC
integer, optional, intent( out) :: RC
```

DESCRIPTION:

User-supplied setservices routine. The register routine sets the subroutines to be called as the init, run, and finalize routines. Note that those are private to the module.

PARAMETERS:

(IMPORT STATE)

```
call MAPL_AddImportSpec(GC, &
    LONG_NAME      = 'eastward_wind_component_on_Cgrid', &
    UNITS          = 'm s-1', &
    SHORT_NAME     = 'UC', &
    DIMS           = MAPL_DIMSHORZVERT, &
    VLOCATION      = MAPL_VLocationCenter, &
                           RC=STATUS )
```

VERIFY_(STATUS)

```
call MAPL_AddImportSpec(GC, &
    LONG_NAME      = 'northward_wind_component_on_Cgrid', &
    UNITS          = 'm s-1', &
    SHORT_NAME     = 'VC', &
    DIMS           = MAPL_DIMSHORZVERT, &
    VLOCATION      = MAPL_VLocationCenter, &
                           RC=STATUS )
```

VERIFY_(STATUS)

```
call MAPL_AddImportSpec(GC, &
    LONG_NAME      = 'eastward_pressure_flux_on_Cgrid', &
    UNITS          = 'm+2 s-1 Pa', &
    SHORT_NAME     = 'MX', &
```

```

DIMS          = MAPL_DIMSHORZVERT,          &
VLOCATION     = MAPL_VLocationCenter,      &
                           RC=STATUS  )

VERIFY_(STATUS)

call MAPL_AddImportSpec(GC,                      &
LONG_NAME      = 'northward_pressure_flux_on_Cgrid', &
UNITS          = 'm+2 s-1 Pa',                  &
SHORT_NAME     = 'MY',                         &
DIMS           = MAPL_DIMSHORZVERT,          &
VLOCATION      = MAPL_VLocationCenter,        &
                           RC=STATUS  )

VERIFY_(STATUS)

call MAPL_AddImportSpec(GC,                      &
LONG_NAME      = 'upward_mass_flux_due_to_dynamics', &
UNITS          = 'kg m-2 s-1',                  &
SHORT_NAME     = 'MZ',                         &
DIMS           = MAPL_DIMSHORZVERT,          &
VLOCATION      = MAPL_VLocationEdge,         &
                           RC=STATUS  )

VERIFY_(STATUS)

call MAPL_AddImportSpec(GC,                      &
LONG_NAME      = 'air_pressure_tendency_due_to_nondynamical_processes', &
UNITS          = 'Pa s-1',                     &
SHORT_NAME     = 'DPEDT',                        &
DIMS           = MAPL_DIMSHORZVERT,          &
VLOCATION      = MAPL_VLocationEdge,         &
                           RC=STATUS  )

VERIFY_(STATUS)

call MAPL_AddImportSpec(GC,                      &
LONG_NAME      = 'initial_pressure_thickness',   &
UNITS          = 'Pa',                          &
SHORT_NAME     = 'DP',                         &
DIMS           = MAPL_DIMSHORZVERT,          &
VLOCATION      = MAPL_VLocationCenter,        &
                           RC=STATUS  )

VERIFY_(STATUS)

call MAPL_AddImportSpec(GC,                      &
SHORT_NAME     = 'TRACERS',                     &
LONG_NAME      = 'tracer_mixing_ratios',       &
units          = 'X',                           &
DIMS           = MAPL_DIMSHORZVERT,          &
VLOCATION      = MAPL_VLocationCenter,        &
DATATYPE       = MAPL_BundleItem,             &

```

```

VERIFY_(STATUS)

call MAPL_AddInternalSpec ( gc,
    SHORT_NAME = 'UC',
    LONG_NAME  = 'eastward_wind',
    UNITS       = 'm s-1',
    PRECISION   = ESMF_KIND_R8,
    DIMS        = MAPL_DimsHorzVert,
    VLOCATION   = MAPL_VLocationCenter,
    RC=STATUS   )
&
VERIFY_(STATUS)

call MAPL_AddInternalSpec ( gc,
    SHORT_NAME = 'VC',
    LONG_NAME  = 'northward_wind',
    UNITS       = 'm s-1',
    PRECISION   = ESMF_KIND_R8,
    DIMS        = MAPL_DimsHorzVert,
    VLOCATION   = MAPL_VLocationCenter,
    RC=STATUS   )
&
VERIFY_(STATUS)

```

1.1.2 Initialize - initialization routine

INTERFACE:

```
subroutine Initialize(GC, IMPORT, EXPORT, CLOCK, RC)
```

ARGUMENTS:

```

type(ESMF_GridComp), intent(inout) :: GC
type(ESMF_State),    intent(inout) :: IMPORT, EXPORT
type(ESMF_Clock),   intent(inout) :: CLOCK
integer, optional,   intent(  out) :: RC

```

DESCRIPTION:

This initialization routine creates the import and export states, as well as the internal state, which is attached to the component. It also determines the distribution (and therefore the grid) and performs allocations of persistent data,

1.1.3 Run - run routine

INTERFACE:

```
subroutine Run(GC, IMPORT, EXPORT, CLOCK, RC)
```

ARGUMENTS:

```
type(ESMF_GridComp), intent(inout) :: GC
type(ESMF_State),   intent(inout) :: IMPORT, EXPORT
type(ESMF_Clock),   intent(inout) :: CLOCK
integer, optional,  intent(out)  :: RC
```

DESCRIPTION:

The Run method advanced the advection one long time step, as specified in the configuration. This may be broken down into a number of internal, small steps, also configurable. Each time step is split into a horizontal advection calculation, which is vertically Lagrangian, and a vertical remapping.

The API for remap requires the edge pressures before and after the horizontal transport (Lagrangian) step. These are computed at the FVAdv precision.

The calculation of PE1 (new pressures on Lagrangian frame) is straightforward. The desired final pressures in the Eulerian frame, PE2, comes from the continuity equation for Eulerian coordinates written for the layer thicknesses $\delta_k dP$:

$$\frac{\partial(\delta_k P)}{\partial t} = -\frac{1}{\Delta x \Delta y} (\delta_x(M_x) + \delta_y(M_y)) - g \delta_z(M_z)$$

This is time split between Lagrangian and vertical remapping processes:

$$\begin{aligned} \left(\frac{\partial(\delta_k P)}{\partial t} \right) &= \left(\frac{\partial(\delta_k P)}{\partial t} \right)_L + \left(\frac{\partial(\delta_k P)}{\partial t} \right)_R, \\ \left(\frac{\partial(\delta_k P)}{\partial t} \right)_L &= -\frac{1}{\Delta x \Delta y} (\delta_x(M_x) + \delta_y(M_y)), \\ \left(\frac{\partial(\delta_k P)}{\partial t} \right)_R &= -g \delta_z(M_z). \end{aligned}$$

In finite difference form, the two time steps are:

$$\begin{aligned} (\delta_k P)^* &= (\delta_k P)^{(old)} - \frac{\Delta t}{\Delta x \Delta y} (\delta_x(M_x) + \delta_y(M_y)), \\ (\delta_k P)^{new} &= (\delta_k P)^* - \Delta t g \delta_z(M_z). \end{aligned}$$

The first update, resulting in $(\delta_k P)^*$ is done in the internal FVadvRun routine, which implements the Lagrangian processes. The PE1 are the edge pressures corresponding to these thicknesses. The second update is done below, while computing the final thicknesses, $(\delta_k P)^{new}$. The PE2 are the edge pressures corresponding to these thicknesses. The vertical remap of the tracers is between PE1 and PE2.

1.1.4 Finalize - user supplied finalize routine

INTERFACE:

```
subroutine Finalize(GC, IMPORT, EXPORT, CLOCK, RC)
```

ARGUMENTS:

```
type(ESMF_GridComp), intent(inout) :: GC
type(ESMF_State),    intent(inout) :: IMPORT, EXPORT
type(ESMF_Clock),    intent(inout) :: CLOCK
integer, optional,   intent(  out) :: RC
```

DESCRIPTION:

Finalize merely destroys the FVadv object that was created in Initialize and releases the space for the persistent data .

1.1.5 TestCase

INTERFACE:

```
subroutine TestCase(gc, import, export, clock, rc)
```

USES:

```
use FVAdvUtilsMod, only : FVAdvUtilsTestCase
implicit none
```

ARGUMENTS:

```
type(ESMF_GridComp), intent(inout) :: gc
type(ESMF_State),    intent(inout) :: import
type(ESMF_State),    intent(inout) :: export
type(ESMF_Clock),    intent(in)    :: clock
integer, intent(out), optional     :: rc
```

DESCRIPTION:

This routine provides a test scenario; usually this routine will not be exercised since another component will provide the import state.

If test data are desired, this routine can be activated by setting the **TEST:** variable in the configuration file to **1** (the default is **0**, or not active).

REVISION HISTORY:

2007-07-09 Sawyer Creation from FVHS example